# 1 Master mind

Il s'agit de programmer une variante du jeu de *master mind*, en devinant des «mots» de $N$ lettres ($N = 5$ ou 6). On considèra pour simplifier qu'un mot est toute combinaison de $N$ lettres minuscules, mais ces mots pourraient aussi être pris dans un dictionnaire.

Dans ce jeu, le joueur $A$ choisit en secret un mot que le joueur $B$ doit deviner en faisant des *propositions* auxquelles le joueur $A$ donne des réponses indiquant

- le nombre $P$ de lettres de la proposition à la même place que dans le mot à deviner,
- et le nombre $p$ de lettres de la proposition qui sont présentes dans le mot à deviner, mais pas à la même place.

Par exemple, mot à deviner = **master**, proposition = **tarot** : la réponse sera $(P = 1, p = 2)$. Noter que le **t** ne compte qu'une seule fois.

Le joueur $B$ a gagné s'il obtient la réponse $(P = N, p = 0)$ en au plus $M$ propositions (par exemple, $M = 2N$).

Le programme devra être capable de jouer le rôle du joueur $A$ et du joueur $B$.

## Rôle du joueur $A$.

C'est la partie facile. Le programme choisit un mot ($N$ lettres au hasard), et se contente de répondre à des propositions entrées au clavier. Il déclarera évidemment que $B$ a perdu s'il n'a pas trouvé après $M$ propositions.

Attention cependant : l'algorithme pour calculer la réponse est moins simple qu'il ne semble.

## Rôle du joueur $B$.

Le programme doit trouver le mot en fonction de réponses données au clavier.

Il existe un algorithme simple pour trouver à coup sûr (si le joueur $A$ a été honnête dans ses réponses). La solution à trouver est (pour 5 lettres par exemple) un mot compris entre **aaaaa** et **zzzzz**, pour lequel il doit y avoir les mêmes réponses aux propositions de $B$ que les réponses de $A$.

Supposons que la première proposition de $B$ soit *aaaaa* et que la réponse de $A$ soit $(P = 1, p = 0)$. Alors, le prochain (dans l'ordre lexicographique) mot possible est *abbbb*. Si $A$ répond $(P = 0, p = 2)$, alors le prochain mot possible sera *baccc*, etc.

En d'autres termes, en partant de la proposition *aaaaa* et de la réponse $(P = 1, p = 0)$, le programme examine *aaaab*, qui ne convient pas car si c'était le mot à deviner, alors la réponse aurait été $(P = 4, p = 0)$, et ainsi de suite jusqu'à *aaaaz*, puis *aaaba*, ...*aazzz*, ...*abaaa*, ..., qui ne conviennent toujours pas, et enfin *abbbb* dont la réponse est compatible avec le mot à deviner. Donc, de manière générale, la prochaîne proposition du programme sera un mot suivant sa dernière proposition, et dont les réponses à toutes ses propositions précédentes sont les mêmes que celles qu'il eues de $A$.

Il est clair que si le programme atteint $zzzzz$ sans avoir trouvé, c'est que le joueur $A$ n'a pas été honnête dans ses réponses.

**Amélioration**

Rechercher dans les mots de $aaaaa$ à $zzzzz$ en suivant l'ordre lexicographique habituel est la meilleure garantie de ne trouver qu'en 26 coups (il suffit que $A$ choisisse $zzzzz$).

Il est donc préférable de commencer avec une proposition dans laquelle il y a peu ou pas de lettres répétées, et qui sont tirées au hasard pour que le comportement du programme ne soit pas humainement prévisible.

Dans le même ordre d'idée, il faut éviter de suivre l'ordre lexicographique habituel car le programme arrive vite à des propositions du genre $kkkkk$. Il est préférable de définir un ordre dans lequel la lettre suivant une lettre $l$ à la $k$-ième position d'un mot est $l + d_k$ *modulo* 26. Il suffit que $d_k$ soit premier avec 26 pour être sûr d'examiner tous les mots possibles avant de revenir au mot initial. Par exemple, les $d_k$ peuvent être respectivement $1, 3, 5, 7, 9$, si bien que le mot suivant $aaaaa$ est $bdfhj$, et celui suivant $uuuuu$ est $vxzbd$.

## Un joueur $A$ tricheur indétectable

Si le programme dans le rôle du joueur $A$ note, comme pour le rôle de $B$, toutes les propositions qui lui ont été faites et les réponses qu'il a données, alors il lui est facile, dès que l'adversaire a trouvé (ou s'approche dangeureusement de la solution), d'essayer de trouver un nouveau mot à deviner qui soit compatible avec toutes les réponses qu'il a données : il suffit d'appliquer le même algorithme que $B$ applique pour trouver sa prochaîne proposition, en partant du mot à trouver initial, et en ne s'avouant vaincu que si l'algorithme redonne ce même mot.

La triche est indétectable, sauf que la réponse prendra plus de temps. Ça se plaide sans problème devant les tribunaux.